YOLO9000: Better, Faster, Stronger

Abstract

- novel and drawn from prior works method により state-of-the-art かつ高速
- "Our joint training allows YOLO9000 to predict detections for object classes that don’t have labelled detection data."
  - これどゆこと？？？
- we validate our model on ImageNet detection task
  - 200 class 中 44 class にしか detection data がない ImageNet detection validation set に対して 19.7mAP
  - detection data がない 156 class に対しても 16.0mAP
- real-time で 9,000 classes 以上を検出可能

1. Introduction

- detection は classification に比べ dataset に制約がある
  - most common classification datasets: 数十万クラス，何百万枚の画像
  - most common detection datasets: 数千クラス，数千から数十万枚の画像
- detection も classification の scale にしたいがラベルづけが大変，当分無理そう
- 既存の large amount of classification data を使って detection system の scope を expand する手法を開発
  - classification の hierarchical (階層的な) 視点を用い，複数のデータセットを統合することに成功した
- detection data からも classification data からも学習を可能にする joint training algorithm を提案
  - classification images を使って detection の性能を向上 (leverage)
- 本論文の構成は以下：
  i. YOLO を improve して YOLOv2 に
  ii. dataset combination method と joint training algorithm の導入

2. Better

機械学習の評価方法

	$y=1$	$y=0$
$\hat{y}=1$	True Positive (TP)	False Positive (FP)
$\hat{y}=0$	False Negative (FN)	True Negative (TN)

- Accuracy: 正解率，$\frac{TP+TF}{TP+FP+FN+TN}$
  - 全体のうちどれだけあってるか
- Precision: 適合率，$\frac{TP}{TP+FP}$
  - positive と予測したもののうちどれだけあってるか (これが高いとがむしゃらに true って言ってることになる)
- Recall: 再現率，$\frac{TP}{TP+FN}$
  - 正しいもののうちどれだけを予測できたか

概要

- YOLO は他の state-of-the-art な手法に比べ，多様な欠点に悩まされてきた
  - localization errors がとても多い
  - recall が小さい (FP は少ないけど FN が多い，つまり見落としてる)
    - 識別率を維持しつつ，recall と localization を improve する方針
- 最近のトレンドはネットワークを深くしたりアンサンブルによりパフォーマンスを上げることだけど，YOLO は速さを維持したいのでそうしなかった

Batch Normalization

- batch normalization を全ての convs につけることで 2% improvement in mAP
- 過学習を避けつつ dropout をなくすことができた
- batch normalization
  - 勾配消失・爆発を防ぐ
  - 今までは活性化関数の変更，weights の初期値の事前学習，lr を小さくする，dropout などの手法により対処してきたがこれらが不要に
  - 共変量シフト (Covariate Shift): 訓練データと予測データの入力の分布に偏りがあること
    - 内部の共変量シフト (Internal Covariate Shift): 隠れ層において層と activation 毎に入力分布が変わること
  - まあ要するに途中で conv の出力とかを batch 単位で正規化すること

High Resolution Classifier

- pre-training では 224 x 224 で行なっていたが，448 x 448 のフルサイズで行なった，10 epochs
- increase almost 4% mAP

Convolutional With Anchor Boxes

- YOLO では FC 層で bounding boxes の座標を得ていたが，Faster R-CNN では hand-picked priors と呼ばれる convs のみの layer で得ている
  - conv layers しか使っていないので，Faster R-CNN の region proposal network (RPN) はoffsets and confidences for anchor boxes を予測する
  - YOLOv2 でもこれを採用
- 変更点は以下
  i. FC 層をなくして anchor boxes を使った
  ii. 解像度を上げるために pooling なくした
  iii. 入力画像を 448 x 448 から 416 x 416 にした
    - 32 の奇数倍にして center cell を一意に定めるため
    - 13 x 13 の feature map を得る
  iv. class prediction を spatial location から切り離し，それぞれの anchor box について class と objectness を予測するようにした
- anchor box の採用による影響
  - accuracy は若干下がった
  - YOLO では 98 boxes しか予測できなかったが，千以上の box について予測できるようになった
    - without anchor box: 69.5mAP with a recall of 81%
    - with anchor box: 69.2mAP with a recall of 88%
  - anchor box is 何?
    - 単純に各 sliding-window に対して複数の scale, aspect ratio の bounding box をやる

Dimension Clusters

- YOLO で anchor boxes を使用することによる2つの問題点のうちの1つ目：box dimensions are hand picked について
  - network は box の adjust を学習することができるが，適切な prior (前例，優先順位) を設定することでその学習をより容易にすることができる
  - prior を人が決定するのではなく，k-means clustering により行う
  - ユークリッド距離によってクラスタリングを行うとでかい box が大きな error を出してしまうので，$d = 1 - \mathrm{IOU}$ として定義した
  - IOU は k と正の相関を持ったが，model complexity と recall とのトレードオフで $k = 5$ とした
  - $k = 5$ で hand-picked な 9 anchor box と同等の性能 (Ave. IOU = 61%)，$k = 9$ では 67.2%

Direct Location Prediction

- YOLO で anchor boxes を使用することによる2つの問題点のうちの2つ目：特に学習初期におけるモデルの不安定性
  - 主に (x, y) を予測するところに起因
  - Region Proposal Networks では t_x, t_y を導入して解決していたが，これは任意の box を出力できる代わりに学習が大変

- 今回は YOLO を踏襲して grid と bounding box の中心を対応づける
- 以下の t_x, t_y, t_w, t_h, t_o を予測する (grid cell の左上を (c_x, c_y) とする)

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
Pr(\text{object}) \times IOU(b, \text{object}) &= \sigma(t_o)
\end{aligned}
$$

- 学習が容易になったので anchor box に比べ 5% の性能上昇

Fine-Grained Features

- 13 x 13 は小さな object には不十分なことがある
  - Faster R-CNN, SSD では複数の scale の feature maps に proposal networks をつないでいたが，YOLO では 26 x 26 の feature map からの passthrough を導入することにより解決する
- passthrough layer では higher resolution layer を lower resolution layer に concat
  - 1つのチャンネルから4つのチャンネルにつなぐ (ResNet に似てる)
    - 26 x 26 x 512 -> 13 x 13 x 2048
- 1% の改善

Multi-Scale Training

- conv layers のみで構成されているのでサイズ不変
- $1/32$ にダウンスケールされるので 10 batchs ごとに一辺の長さを 32 ずつ $\{320, 352, \ldots, 608\}$ と random に変化させた
- input resolution を変えることで速さと正確性の trade off ができるお

Further Experiments

- PASCAL VOC 2007: high resolution YOLOv2 が最強
- PASCAL VOC 2012: SSD 512 が最強

Faster

- 多くの frameworks は VGG-16 を base feature extractor として使っているが，224 x 224 の画像1枚に対して 30 billion もの浮動小数点演算を要求するので非効率
  - YOLOのカスタムモデルでは 224 x 224 の画像に対して 8.5 billion
  - ImageNet での性能を比較，VGG-16 の 90.0% に対して 88.0% を記録

Training for Classification

- Darknet-19 を 224 x 224 で pre-train したのちに 448 x 448 で fine-tuning
- 1000 classes, 160 epochs using SGD
- data augumentation

Training for Detection

- Darknet-19 の last conv を detection 用の 3 x 3, 1 x 1 の conv に交換し，passthrough leyer を追加
- 160 epochs
- data augumentation with the same way as SSD

Stronger

- classification と detection を jointly に学習するための mechanism を propose
  - クラスラベルのみのデータも detection の学習に使える
- training 中は classification と detection の datasets を mix して使う

    - detection 用の data に対しては architecture 全体の loss function で backropagate
    - classification 用の data に対しては loss fuction のうち識別に関わるの部分のみ backropagate
        - But how?
- 複数の datasets を使うためにはいくつか解決しなければならない問題がある
    - detection のクラスは少なく classification のクラスは多い
        - COCO では "dog" のみでも ImageNet では "Norfolk terrier", "Yorkshire terrier", and "Bedlington terrier" 等 100 種類もある
    - 多くの classification model で用いられる softmax は mutual exclusive (互いに排反) であるが，例えば "dog" と "Norfolk terrier" は排反ではない
- この問題を解決するために multi-label model を使用

Hierarchical Classification

- ImageNet のラベルは WordNet からもって来ている
    - WordNet 内の単語はグラフ構造をもつ
        - "Norfolk terrier" < "terrier" < "hunting dog" < "dog" < "canine"
    - この階層構造を採用する
- WordNet 内のグラフ構造は有向グラフであり，木構造ではない
    - "dog" が "canine" でもあり "domstic animal" でもあるように言語は複雑だから
    - graph 構造の代わりに，hierarchical tree を作成した
    - ImageNet 内の単語について，WordNet のグラフを用いて root node への path を特定
        - 多くの場合 path は1つのみ
    - そのように構築されたグラフに path を追加したり取り除いたりすることで木を最小化した
    - このようにして出来上がった木を WordTree と呼ぶ
- 各ノードについて条件付き確率 (ex: $Pr(\text{Norfolk teriier}|\text{terrier})$) を予測する

$$Pr(\text{Norfolk teriier}) = Pr(\text{Norfolk teriier}|\text{terrier}) \times Pr(\text{teriier}|\text{hunting dog}) \times \cdots \times Pr(\text{mammal}|\text{animal}) \times Pr(\text{animal}|\text{p}$$

- WordTree を構築するために ImageNet の 1000 クラスから 1369 まで中間ノードを追加
    - "Norfolk terrier" に対しては "dog", "mammal" も予測するようにした
    - クラス増やしたけど性能は落ちなかった
    - 学習していない犬を入力すると，"dog" の値は高いが下位の単語の値は全て低くなる
- detection の際には，detector が生成した bounding box の予測する tree of probability 中を，最も confidence score が高い path を選択するように探索し，score threshold を超えたらそのクラスを出力とする

Joint Classification and Detection

- COCO に ImageNet の top 9000 クラスを追加して 9418 クラスの WordTree を作成
    - ImageNet の方がデータ数が多いので COCO からは oversampling した，結果 ImageNet:COCO = 4:1
- この WordNet で学習した YOLOv2 が YOLO9000
    - ただし，anchor box の k-means clustering は $k = 3$ に変更
- 学習について
    - When YOLOv2 sees a detection image
        - 普通にバックプロパゲート
        - classification についてはより上位の単語まで学習
            - "dog" でポシャっても "German Shepherd” versus “Golden Retriever" については error を与えない
    - When YOLOv2 sees a classification image
        - classification loss のみバックプロパゲート
            - そのクラスに対して最も高い score を返す bounding box を特定し，その predicted tree について loss を計算
        - "We also assume that the predicted box overlaps what would be the ground truth label by at least .3 IOU" が何言ってるかわからん